Análisis sobre enfoques de Deep Learning para la generación de resúmenes

Trabajo de Fin de Grado
Curso 2019–2020

Autores
Álvaro Pascual Núñez

Director
Alberto Díaz Esteban

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Análisis sobre enfoques de Deep Learning para la generación de resúmenes

Trabajo de Fin de Grado en Ingeniería Informática
Departamento de Ingeniería de Software e Inteligencia Artificial

Autores
Álvaro Pascual Núñez

Director
Alberto Díaz Esteban

Convocatoria: *Junio 2020*
Calificación:

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

25 de junio de 2020

Agradecimientos

Querría agradecer a todas aquellas personas que han estado apoyándome durante todo este año. Gracias a todos los familiares y amigos que tanto ánimo me han dado en los momentos más difíciles y que han soportado mis innumerables charlas sobre lo que estaba haciendo cada vez que me preguntaban. Muchas gracias también al director de este proyecto, Alberto Díaz Esteban, que me ha acompañado durante toda esta aventura tan interesante. De verdad, muchas gracias por toda la ayuda y consejos.

Resumen

En la actualidad, la cantidad de información de la que disponemos en Internet es inmensa. Que todo el mundo sea capaz de acceder y extraer de ella lo más relevante y comprender todo de forma clara es algo que se ha estado investigando durante las últimas décadas. Numerosas implementaciones han surgido en proyectos que se desarrollaban bajo las estructuras y mecanismos disponibles en el campo del *Deep Learning*. Cada modelo cuenta con características particulares e intentando resolver problemas diferentes, pero siempre con un objetivo común: resumir de la forma más eficaz cualquier texto.

Todo el código fuente referente a este proyecto puede encontrarse en:

`https://github.com/NILGroup/TFG-1920-Resumenes`

Palabras clave: resumen, información, deep Learning, NLP, redes neuronales, RNN, seq2seq, encoder-decoder, word embedding, mecanismo de atención, cornerstone, pointer generator, reinforcement learning, parendizaje por esfuerzo, policy gradient, Tensorflow, Python.

Abstract

Nowadays, the amount of information available on the Internet is vast. Something that has been researched for the last decades is the opportunity for everyone to be able to access that information and be able to extract the most relevant part of it and clearly understand it. Numerous implementations have emerged in projects developed were based on the structures and mechanisms available on the Deep Learning field. Every model has its own characteristics and tries to solve varios problems, but those models always follow one unanimous objective: summarize any text in the most effective way.

All the source code referred in the project can be found here: `https://github.com/NILGroup/TFG-1920-Resumenes`

Keywords: summary, information, deep learning, NLP, neural networks, RNN, seq2seq, encoder-decoder, word embedding, attention mechanism, cornerstone, pointer generator, reinforcement learning, policy gradient, Tensorflow, Python.

Índice

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

La información es uno de los recursos más importantes de la actualidad. Continuamente encontramos cantidad de información nueva, desde artículos en los periódicos hasta *posts* en *foros* de Internet. Además, ésta proviene de cualquier rincón del mundo. Todo se lo debemos a la globalización y a los grandes avances en la tecnología. Pero incluso con la sobrecarga de noticias a la que estamos sometidos, sentimos la necesidad de mantenernos al día sobre lo más relevante. A través de nuestros computadores y *smartphones* podemos visitar *portales* y consultar lo qué sea y dónde sea con una rapidez abrumadora. Esto significa que la información está al alcance de nuestra mano con gran facilidad, es decir, es accesible para todos.

Muchos son los usuarios y expertos que han decidido embarcarse en la búsqueda de una solución con la que poder reducir esta gran cantidad de información y condensarla de manera eficaz. Así, no solo se asegura la accesibilidad, sino que también se facilita la comprensión de ésta. Con esto en mente, se han desarrollado durante las dos últimas décadas numerosos proyectos que pretenden lograr la producción de resúmenes automáticos que, además, se asemejen a los que los humanos podemos generar.

1.1. Objetivo

El objetivo de este trabajo es ofrecer un análisis de algunos de los modelos que han surgido en varios proyectos desarrollados en los últimos años. Con esto se pretende dar una visión general del alcance del *Deep Learning* para

la generación de resúmenes automáticos y las diferencias que presentan sus estructuras, tanto en su enfoque, su implementación y la manera en la que resuelven problemas ocasionados por los algoritmos.

El análisis de los modelos consistirá en explicar la base sobre las que sus estructuras han sido construidas y las particulares que pueden presentar. También se configurarán y realizarán experimentos, modificando y probando diferentes características y parámetros de los modelos, para comparar resultados. Con los resultados obtenidos en las distintas pruebas, se realizarán comparaciones globales en cuanto a métricas y matices generales lingüísticos, y además se mostrarán algunos ejemplos en detalle.

1.2. Estructura del documento

Los diferentes capítulos que componen este documento son:

- **Capítulo 1, Introducción.** Motivación y objetivo del proyecto.
- **Capítulo 2, Deep Learning y el resumen.** Explicación sobre las estructuras y componentes para la generación de resúmenes automáticos a través de *Deep Learning*.
- **Capítulo 3, Modelos en detalle.** Explicación en profundidad de la funcionalidad de los códigos usados.
- **Capítulo 4, Resultados.** Análisis de resultados de los modelos.
- **Capítulo 5, Conclusión y trabajo futuro.** Conclusiones y funcionalidades posibles a añadir.

CHAPTER 1

Introduction

Information is one of the most valuable resources in today's world. We constantly find a great amount of new information from newspapers' articles to posts on Internet forums. Moreover, all of these info comes from every corner in the world. And we owe all of this to globalisation and the breakthroughs in technology over the last decades. Yet with this news overload to which we have been exposed to, we feel the necessity to keep ourselves updated on the most relevant matters. Through our computers and smartphones, we are able to navigate through websites and seek whatever and wherever with an astonishing quickness.The meaning behind this leads us to have any fact and data within our hands' reach. And what's even more important, information becomes accessible for everyone.

Many users and experts have decided to get involved in the search for a solution which could help to reduce the great amount of information available and efficiently condense it. This way not only accessibility is guaranteed, but also facilitates the understanding of this info. Bearing this in mind, numerous projects have been developed during the last two decades. These projects aim to achieve the production of summaries that resemble those that humans can create.

1.1. Objective

The main purpose while working in this project is to offer an analysis of some of the models that have emerged and developed over the years. With

this, we intended to give a general vision or overview of *Deep Learning's* range and extent, and the different features that those models have in their unique view, their implementation and the way the solve issues that algorithms can cause.

Models analysis will consist of explaining the basis on which the implementations structures have been built and the particularities they might present. Experiments will also set up and carried out, modifying and testing different characteristics and parameters of the models, to compare results. Later with the results obtained in the different tests, global comparisons will be made in terms of metrics and general linguistic nuances, and some detailed examples will also be shown.

1.2. Document structure

The content of the different chapters in this document are:

- **Chapter 1, Introduction.** Motivation and objective.
- **Chapter 2, Deep Learning and summarization.** Structure and model explanation about how neural networks solve the summarization task.
- **Chapter 3, Models in detail.** Explanation in depth of the models' code used.
- **Chapter 4, Results.** Analysis of models results.
- **Capítulo 5, Conclusion anf further work.** Reached conclusions and possible funcionalities to add.

CAPÍTULO 2

Deep Learning y el resumen

Como ya se ha comentado anteriormente, en la actualidad estamos saturados de tanta información. El poder simplificar toda ésta y reducirla a lo esencial se ha convertido en algo totalmente necesario. Pero antes de explicar como tratan de solventar este problema, tenemos que explicar un poco en que consiste resumir y porqué hacerlo de una manera u otra.

2.1. El resumen

Todos alguna vez hemos tenido que enfrentarnos a la tarea de resumir o simplificar un texto. Cada uno tiene una técnica, y ésta se ajusta más o menos a la cantidad de palabras que cada uno puede manejar. También hay que tener en cuenta la complejidad de la información a resumir: si contamos con un texto con un vocabulario o ideas difíciles de entender, puede que necesitemos de más tiempo para leer e interiorizar los conceptos más importantes.

De entre todos los métodos disponibles, los más usados son el *extractivo* y el *abstractivo*. El método *extractivo* intenta seleccionar pequeños trozos del texto y palabras aisladas para poder juntarlas y crear el resumen. Ésta es una de las maneras más extendidas de afrontar los resúmenes, tanto por los humanos como por las máquinas que los programadores han creado. Reordenar las palabras para formar un nuevo y reducido texto es muy conveniente, fácil en cierta manera y garantiza concordancia con el original. Pero al final, esta forma de resumir, la *extractiva*, no hace justicia a esa idea de entender

y resumir. No es humano.

Aquí es donde entra en juego el método *abstractivo*, el cúal se basa en técnicas de generación del lenguaje natural para construir frases. Esto es un proceso más humano, pero también mucho más complicado en comparación al anterior. Dotar a una máquina de la suficiente capacidad como para discernir entre ideas y conceptos con relevancia a lo largo de artículos, o incluso novelas y cuentos, es algo mucho más costoso.

2.2. El modelo

A pesar de que las personas ya sabemos lo que significa resumir, una máquina no. Para lograr esto deberemos transmitir esas nociones, extraer o abstraer ideas, a las máquinas e implementarlas.

Siempre que nos disponemos a resumir, lo hacemos partiendo de un contexto. Este contexto es algo innato para nosotros los humanos. Comprende desde el lenguaje en el que nos expresamos, el significado que le damos al vocabulario y hasta las ideas que formamos entorno a él. Dentro del cuál, y partiendo de un texto original, produciremos otro texto, esta vez más corto, que preservará el sentido y los conceptos transmitidos por el primero.

Siguiendo esta idea se definirá cual será la base del modelo. Ésta no es más que la existencia de una relación entre el texto original y el producido, entre los elementos de uno y los del otro. El objetivo entonces, será implementar una función que nos ayude a encontrar los parámetros con los que computar dichas relaciones.

2.3. *Tokens* y *Word Embeddings*

Todo texto puede ser interpretado como un conjunto, como una secuencia de elementos. Estos elementos son cada una de las palabras y signos de puntuación que componen un texto. A cada uno de éstos se le denominará *token*, y se le asignará un índice que representará su posición con respecto a una lista. A esta lista se la conoce como vocabulario, o diccionario, pues contiene todas las palabras que conforman nuestro contexto.

El vocabulario que se acaba de definir, se compondrá de todos los *tokens* que clasifiquemos. Esta técnica, conocida en el mundo del *Deep Learning* (DL) como *1-of-K coding*, nos ayuda a marcar cada posición del vector con ceros, excepto aquella que ocupa la palabra dentro del vocabulario [Tabla 2.1].

El hecho de haber ordenado las palabras no ofrece la relación entre ellas, ni si dependen unas de otras. Para solucionar esto recurrimos a los *word*

Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
$\begin{bmatrix}0\\0\\0\\0\\\vdots\\1\\\vdots\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\0\\0\\0\\\vdots\\1\\\vdots\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\\vdots\\1\\\vdots\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\0\\0\\0\\\vdots\\1\\\vdots\\0\end{bmatrix}$	$\begin{bmatrix}0\\\vdots\\1\\\vdots\\0\\0\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\0\\0\\\vdots\\1\\\vdots\\0\\0\end{bmatrix}$

Tabla 2.1: Ejemplo de los vectores 1-of-K.

embeddings [10], que son funciones que nos ayudan a mapear palabras en matrices, pudiendo así establecer esas relaciones.

Colocando a modo de columnas los vectores obtenidos con el sistema *1-of-K coding* en la matriz, las filas serán todos aquellos conceptos con los que identificamos las palabras como su información semántica o sintáctica [Tabla 2.2]. Todas estas características e información las establecerá por si sólo el modelo, aprendiendo tal y como lo hace el cerebro humano. Por lo que cuanto mejor sea la representación de esos vectores, mejor serán los resúmenes producidos.

	Man	Woman	King	Queen	Apple	Orange
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

Tabla 2.2: Ejemplo de *word embedding*.

Pero también el modelo debe ir un paso más allá. No solo aprender las características que definen una palabra o las relaciones entre las palabras de una secuencia, sino poder establecer la conexión entre las secuencias origen y las resultado. El método que nos permitirá estos son las *RNN (Recurrent Neural Networks) sequence-to-sequence.*

2.4. Redes neuronales

Las redes neuronales son la herramienta perfecta para evaluar y aprender las relaciones entre secuencias. Estas redes son semejantes a las que componen nuestro cerebro y siguen un proceso similar a la hora de trabajar. Cada celda, o neurona, computa los valores que recibe por un extremo, con una función y unos pesos determinados, y pasa al siguiente nivel de celdas su resultado. Aunque la forma más tradicional de representar estas redes es con un flujo que parte de atrás hacia adelante, conocido como *feedforward*, en el caso específico de resumir encontramos el inconveniente de no poder ajustar a una longitud fija el texto que resulta. Para solucionar esto, se recurrirá a las RNN (*Recurrent Neural Networks*).

2.4.1. Redes Neuronales Recurrentes

Puesto que los resúmenes no tienen una longitud fija hasta que se generan, no se obtienen siempre k palabras resultantes de n entrantes. Esto incluso, nos lleva a esa idea anterior de que la información fluye en un solo sentido. Aunque la etapa anterior ofrece datos útiles para el cómputo, también se debe prestar atención a palabras que hayan aparecido mucho antes o que lleguen a aparecer más tarde. Por lo que necesitamos que haya retroalimentación entre los distintos niveles de la red, es decir, que sea bidireccional.

Otra de las cosas que añadir a estas redes neuronales recurrentes es memoria. Poder mantener guardadas dependencias entre palabras, o mejor dicho, entre las ideas que representan, va a ser de gran ayuda para conseguir realizar resúmenes *abstractivos*. Un tipo concreto de esta clase de redes son las *sequence-to-sequence*. Estas redes de aprendizaje dedicadas al entrenamiento de modelos convierten secuencias de un dominio a otro. Uno de los escenarios más comunes son las traducciones de una lengua a otra, pero se pueden utilizar en cualquier ámbito en el que se procese y genere información de nuestro lenguaje. Estas redes recurren a las unidades GRU ó LSTMs, capaces de guardar en ellas información relevante, incluso en series de eventos con intervalos de duración desconocida. Sabiendo lo importantes que pueden llegar a ser las unidades LSTMs, se detallará un poco más su funcionamiento.

2.4.2. Celdas LSTMs

Las LSTM, o *Long Short Term Memory networks* [6], proporcionan a la red la capacidad de retención de conceptos. El ser humano no empieza a pensar de nuevo cada vez que encuentra una nueva palabra. Según lee, entiende el significado de una palabra en base a las anteriores y al contexto en el que se encuentra. Este es el cometido de las celdas LSTM.

Las redes que componen estas celdas se diferencian fundamentalmente en la estructura en forma de cadena que presentan. Mientras que las RNN normales simplemente computan la entrada y la salida de la anterior celda para producir su resultado [Figura 2.1], las redes de celdas LSTM son algo más complejas [Figura 2.2].

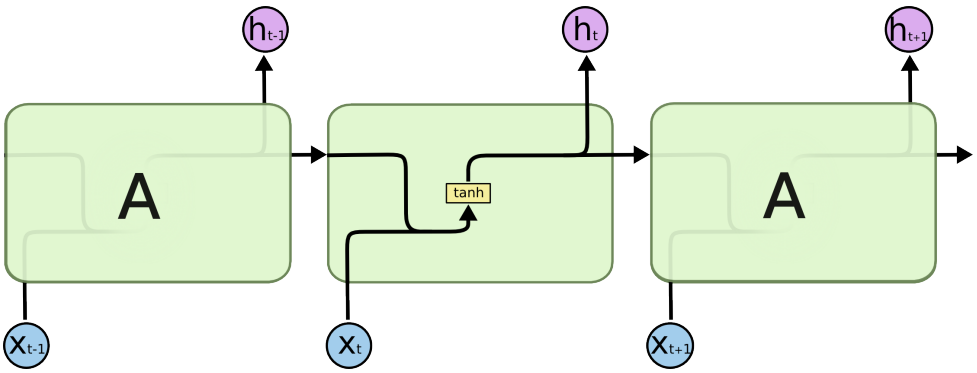

Figura 2.1: Estructura de una red neuronal recurrente fundamental [11].

Lo primero que harán las LSTM es decidir que información mantendrán o desecharán a lo largo de la cadena. A través de una función sigmoide, se calculará entre 0 y 1 la decisión partiendo del *input* y otros datos de género, número o propiedad que la palabra debe cumplir.

Después se deberá decidir que información pasará a la siguiente celda de la red. Aquí, otra función sigmoide llamada *input gate layer*, actualizará unos valores concretos. Además, otra capa empleará una función *tanh* para crear un vector con posibles propiedades candidatas a recordar en futuras celdas.

A continuación se modificará el valor que se recibió desde la celda anterior. Se multiplicará por el valor decidido a olvidar y se le añadirá el producto de los nuevos valores candidatos y el nivel de actualización de los mismos.

Por último se definirá la salida. Análogo al primer paso, la función sigmoide decidirá que valores serán nuestra salida. Luego se normalizará este cálculo con *tanh* para acotar entre -1 y 1.

2.4.3. Estructura *Encoder-Decoder*

Habiendo definido la manera en la que se van a clasificar las palabras, como van a aprenderse las relaciones entre ellas, y las celdas con las que realizar todos los cálculos, solo falta determinar la estructura en las que las redes bidireccionales se van a organizar.

La estructura sobre la que el modelo va a trabajar consta de dos partes: *encoder* y *decoder* [Figura 2.3].

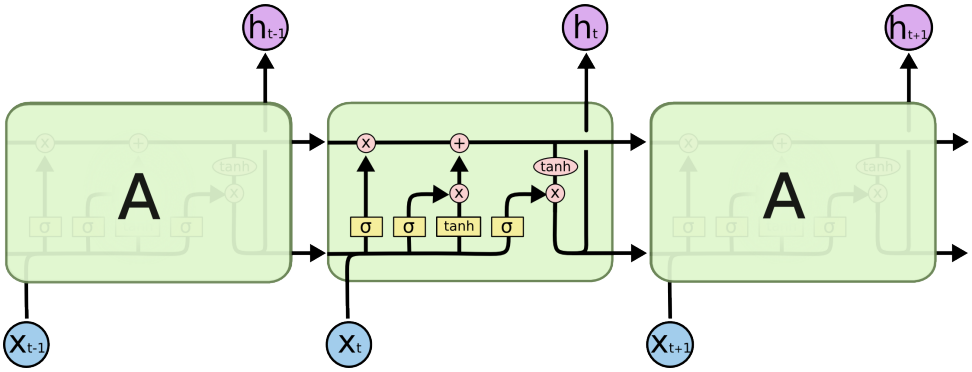

Figura 2.2: Estructura de una RNN compuesta por unidades LSTMs [11].

El proceso que seguirán los modelos para generar los resúmenes comienza al recoger y clasificar las palabras. A cada paso, las distintas celdas del *encoder* clasificarán en las distintas *tokens* dentro del contexto, adquiriendo a cada paso, una mejor representación de los conceptos que han ido procesando.

Tras esto, el *decoder* producirá las nuevas palabras que resultantes. Estas palabras habrán sido calculadas teniendo en cuenta las inmediatamente anteriores en la secuencia y el vector contexto generado en el *encoder*, para lograr maximizar la relación entre el original y el resultado.

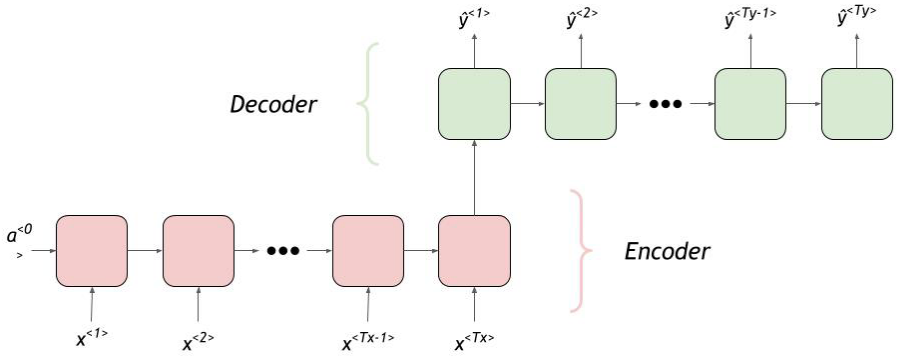

Figura 2.3: Estructura *encoder-decoder*.

Por explicarlo de forma más simple: mientras el *encoder* recoge y clasifica las secuencias de elementos, el *decoder* genera nuevas secuencias según los parámetros aprendidos.

2.4.4. Mecanismo de atención

Algo tan especialmente importante como la relevancia de una palabra y de su dependencia con otras, como ya se ha explicado anteriormente, es el principal foco a la hora de dotar a los resúmenes de ese carácter abstracto. Gracias a la estructura bidireccional de la red, ahora se cuenta con dos vectores que recorren en ambos sentidos el texto. Si entonces se presta atención y se guardan las palabras que alojan la idea principal a transmitir, se logrará mayor precisión a la hora de producir el resumen. De esto se encargará el *encoder*, que en cada paso revisará en qué debe fijarse el *decoder* para generar la nueva palabra.

Para conseguir dotar a los resúmenes de un carácter humano, hay que enseñar a los modelos a prestar atención a varias palabras al mismo tiempo y no a todo el texto. La importancia de una palabra puede venir de su relevancia dentro del contexto, o de su dependencia con otras cercanas a ella.A esto se le conoce como *mecanismo de atención* [4][15]. Gracias a la estructura bidireccional de la red, ahora se recorre en ambos sentidos el texto pudiendo así tener en cuenta las palabras que rodean a la que en cada paso se está observando. Esto ayudará a calcular el vector contexto, que reflejará la cantidad de atención que se le dará a cada palabra.

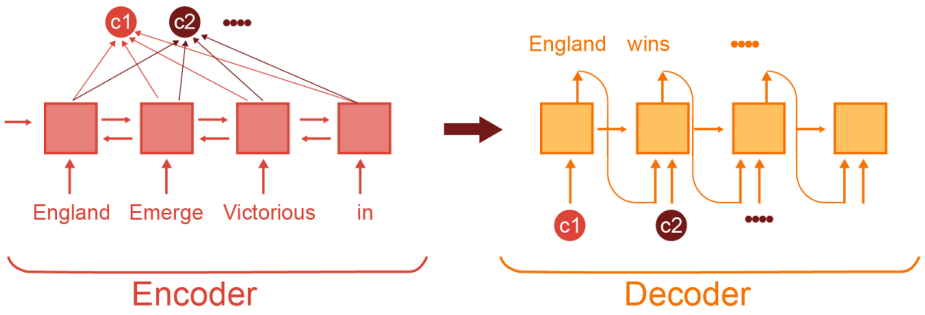

Figura 2.4: Vector contexto formado por las distribuciones de atención de cada palabra [16] .

El vector de contexto se calculará a partir de la atención que recibe a través de cada una de las celdas de ambos sentidos de la red del *encoder*. Estas atenciones se sumarán teniendo en cuenta sus pesos. Luego pasarán por una función que regule su valor, como la función *softmax*. El resultado de este cálculo será entonces introducido en la celda del *decoder* en ese paso, y junto al estado de la celda anterior en la red servirán para producir la nueva salida.

2.5. Evaluación

Para evaluar los resultados obtenidos con cada uno de los modelos entrenados se utilizará ROUGE [8]. Contaremos con dos implementaciones: la implementación en lenguaje *perl* y una implementación completamente en lenguaje *python* . Este conjunto de métricas se emplean para evaluar programas de *Natural Language Processing* (NLP) tales como la generación de resúmenes automáticos.

ROUGE no analiza si el resumen producido es coherente o no, simplemente indica cuantas palabras, o *n-gramas*, coinciden entre el texto de referencia y el generado. Entre las diferentes medidas que ofrece este programa, se van a tener en cuenta tres concretas:

- **ROUGE-1** : medirá el grado en el que los *unigramas* se repiten en ambos textos.
- **ROUGE-2** : medirá el grado en el que los *bigramas* coinciden entre ambos textos.
- **ROUGE-L** : medirá las secuencias más largas coincidentes. No hace falta que las palabras, los distintos *gramas*, se hallen en el mismo orden. Además no hace falta predefinir la longitud de las secuencias a comparar, pues se encargará de buscar aquellas más largas.

Cada una de estas medidas realiza las siguientes mediciones sobre las secuencias de diferentes tamaños que analizan: la precisión, la exhaustividad y la medida-F.

- **Precisión (ROUGE-P)** : marca el grado de precisión, de exactitud del resumen generado. En otras palabras, cuanta información de la seleccionada es relevante y necesaria.
- **Exhaustividad (ROUGE-R)** : llamada *recall* en inglés, que indica con que probabilidad las palabras o *n-gramas* del texto de referencia se han conseguido o recuperado en el texto resumen producido. Es decir, cuantos elementos relevantes han sido seleccionados.
- **Medida-F (ROUGE-F)** : el *F-measure* o *F1 score*, es la media harmónica entre los parámetros de precisión y *recall*. Esta medida es considerada la más indicativa de las tres, pues ofrece un balance entre las dos anteriores.

CAPÍTULO 3

Modelos en detalle

Como el Procesamiento del Lenguaje Natural (*NLP*) es algo que el mundo del *Deep Learning* lleva estudiando durante años, lo conveniente sería analizar distintas maneras de afrontar la tarea del resumen, y así poder observar la precisión y comprensión de los textos producidos por los modelos.

Los diferentes métodos existentes para resumir, el *extractivo* y el *abstractivo*, han dado lugar a variedad de maneras de implementar esto. Además, conforme el campo de la inteligencia artificial y la tecnología sobre la que funcionan ha ido mejorando, nuevos enfoques y procedimientos han ido proporcionando sobresalientes resultados. Uno de los proyectos más conocidos fue el desarrollado por *Google*, el *textSum*[1] *(Text summarization with TensorFlow)*. El objetivo principal del equipo de *Google Brain* era el dotar a las máquinas de capacidad de comprensión y así ayudar a simplificar la gran cantidad de información disponible en Internet. Esto, por supuesto, no es algo fácil de hacer ya que la tarea requiere mayor potencia de cómputo para los ordenadores según la longitud de los textos aumentaba. En 2016, decidieron convertir su código en *open-source* para que así la comunidad siguiera adelante con ello y desarrollara e innovara partiendo de una base completamente puntera.

Para aprender como funcionan los distintos tipos de modelos, se han utilizado dos códigos diferentes con pequeñas variantes en cuanto a lo que pretenden conseguir resumir y la manera en la que sus creadores abordaron la implementación.

[1] Proyecto textSum de Google: `https://ai.googleblog.com/2016/08/text-summarization-with-tensorflow.html`

3.1. Generador de titulares: *Cornerstone seq2seq*

Basándose en las librerías que se encuentran dentro del paquete TensorFlow, Lee Dong-Jun[2] creó en 2018 una versión muy eficiente y simplificada para conseguir resumir textos.El modelo implementado tiene una estructura de *encoder-decoder* con mecanismo de atención.

Para que el *encoder* realice de forma correcta la tarea para la que se le ha preparado, debe recibir la información bien estructurada. Para ello se utilizan los vectores que conforman los *word-embeddings*. La herramienta *GloVe* [13]es un algoritmo de aprendizaje no supervisado que se encargará de crear vectores que representan las palabras de un texto, los *token*. Este además ofrece modelos pre-entrenados con esos vectores y vocabularios ya formados y listos para su uso.

La estructura de la red neuronal sobre la que está organizado este modelo tiene una forma particular: es una red bidireccional apilada [Figure 3.1]. Que cada capa de neuronas este organizada en grupos apilados, formados a su vez tanto por las celdas *forward* como por las *backward*, significa que los resultados de sus iteraciones se concentrarán como *inputs* del siguiente nivel de celdas. De esta manera, las redes de ambas direcciones reciben la misma entrada.

Una vez clasificadas las palabras, es hora de comenzar a generar las palabras del texto resumen. Pero antes debemos diferenciar dos fases: entrenamiento y generación. A pesar de que ambas partes conforman la red del *decoder*, la manera de procesar y calcular son diferentes, mostrando resultados más óptimos en distintas tareas. El entrenamiento se lleva a cabo con un *decoder básico ('BasicDecoder'según la nomenclatura de Tensorflow)*. Este tiene las estructuras y las propiedades de las capas de celdas de *Keras*, que siguen en su base la organización de las LTSMs. El objetivo de este entrenamiento es lograr que los parámetros maximicen la probabilidad de certeza con la que se producirá el resumen del texto original.

En línea con el objetivo de maximizar la probabilidad de que una cierta palabra sea la correcta, se utilizará otro tipo de red para generar o "deducir" el resumen. Ésta será el *Beam Search Decoder*. En lugar de elegir la siguiente palabra que con mayor probabilidad vaya a formar parte del resultado, el *beam search* tendrá en cuenta un rango más amplio: guardando los posibles valores que se pueden llegar a generar en siguientes pasos. Esto significa que, en cada paso, la búsqueda se amplía a una secuencia de mayor tamaño que una sola palabra. Así cada uno de los valores candidatos, considerados junto a las secuencias de palabras que les suceden, van siendo examinados y escogidos uno a uno, repitiendo el proceso para la siguiente palabra a generar.

[2]Repositorio en GitHub con el código desarrollado por Lee Dong-jun: `https://github.com/dongjun-Lee/text-summarization-tensorflow`

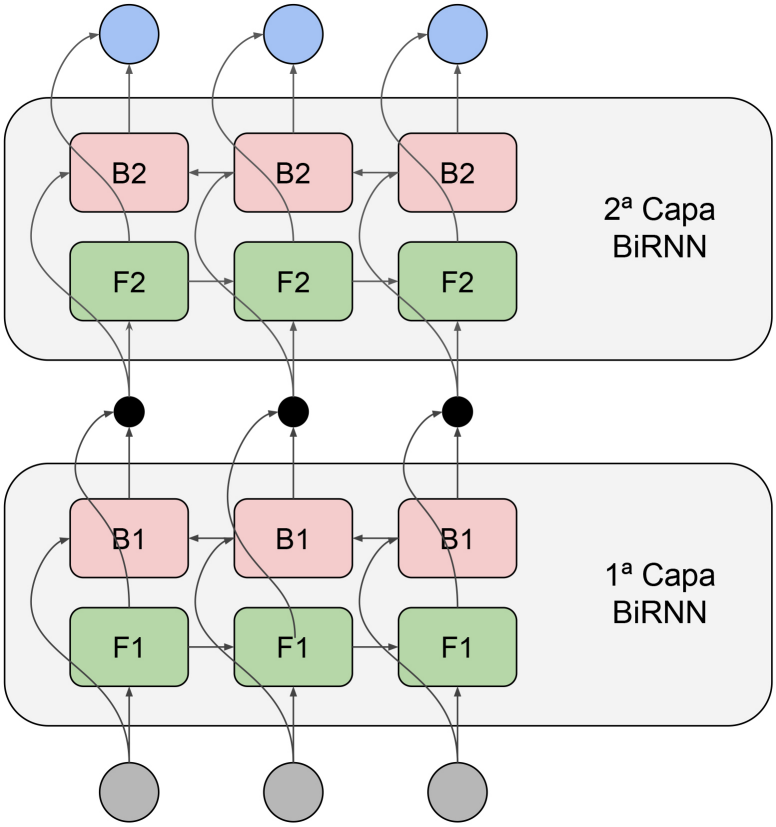

Figura 3.1: Estructura de red neuronal recurrente bidireccional.

Lo último a definir en este modelo es el mecanismo de atención empleado. El mecanismo usado fue desarrollado por Dzmitry Bahdanau[1], y se le conoce comúnmente como *Atención Aditiva* de Bahdanau [9].

Podemos ver que el mecanismo de atención está integrado en la estructura de nuestro modelo [Figura 3.2]. En cada paso, en el cual se procesará una entrada nueva, los nodos de la red *encoder* computarán la información para trasmitirla a través de la red. Además, siendo ésta una de las funciones principales de este mecanismo, guardará esos resultados obtenidos en cada paso para poder más tarde utilizarlos. Tras haber analizado los distintos *token*, los datos pasarán a la segunda parte de la estructura, el *decoder*.

Aquí, el resultado final de la red anterior y esos estados guardados servirán para ajustar con mayor precisión los nuevos resultados de esta red. Prestando atención a estos datos y los valores de las celdas previas en el *decoder* se deberá calcular la probabilidad con la que una palabra formará

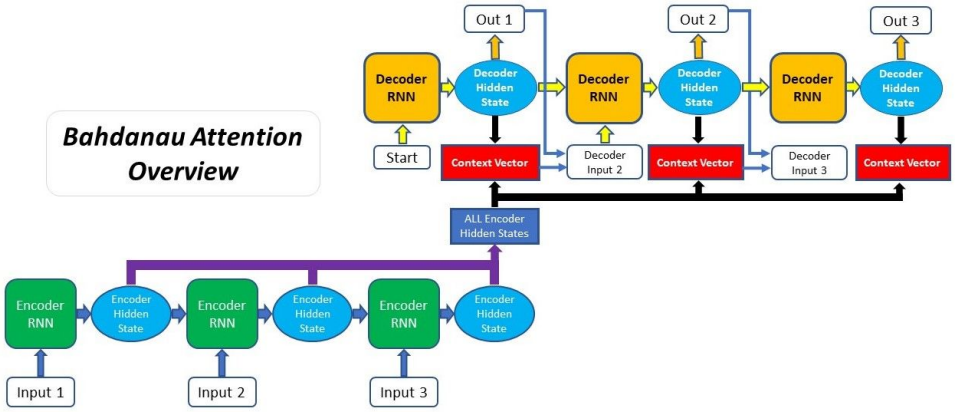

Figura 3.2: Vista global del modelo de Atención *seq2seq* de Bahdanau [9].

parte del resumen. Para ello se combinarán los valores previamente especificados con una función de activación, obteniendo así un vector que sufrirá una transformación (aplicando una función *softmax* que acote los valores entre 0 y 1) para establecer el peso que se le deberá prestar a cada elemento del vector. Estos pesos darán lugar a un vector contexto al aplicarlos sobre los resultados hallados en el *encoder*.

En última estancia, y como ya se ha anotado anteriormente, los nodos de la red recibirán el estado de las celdas que le preceden y el vector contexto que ha sido calculado con respecto a los valores de entrada. La salida final de cada nodo será la nueva palabra del resumen.

3.2. Generador de resúmenes: *Pointer Generator*

Tras la salida de *textSum* como proyecto de código abierto, Abigail See desarrolló un código que se basaba en este. En este proyecto, como indica en su artículo [14], Abigail buscó solucionar algunos de los problemas que presentaban en aquel momento los modelos RNN sobre la generación de resúmenes.

El *Pointer Generator* cuenta con muchas de las estructuras que están presentes en el *Cornerstone seq2seq* (3.1). Pero en este caso, a diferencia del anterior modelo, se han utilizado diferentes organizaciones y técnicas para implementar sus redes y la atención que éstas procesan. Se cuentan con dos tipos de mecanismos de atención diferentes: las redes *pointer-generator*, que usan la distribución de Bahdanau [1] para copiar palabras apuntándolas directamente desde el texto de entrada; y un mecanismo de *cobertura*, que llevará una cuenta de los términos de la entrada a los que ya se han atendido en los anteriores pasos del *decoder*.

Los dos problemas principales a los que hasta la fecha las RNN se enfrentaban, y que estas nuevas técnicas pretenden resolver, eran: la repetición de palabras en el texto generado, y la imprecisión a la hora de mantener detalles de hechos ocurridos. Las repeticiones de palabras se deben a la confianza que el *decoder* otorga a los valores inmediatos de entrada, en lugar de al estado general de la red que podría contener información relevante anterior. También se encontró que los modelos no eran capaces de asegurar la representación de elementos importantes del texto original debido a los *word-embeddings*. Las soluciones que surgieron para resolver estos problemas son la base sobre la que se fundamentó el código[3].

La solución ofrecida para la imprecisión al copiar y mantener conceptos fue el uso de una *pointer-generator network*. Esta clase de red es capaz de elegir si copiar una palabra puntual que tenga cierta relevancia o que sea importante aunque raramente usada, o generar las nuevas palabras siguiendo el vocabulario. Para entender mejor el funcionamiento de este tipo de red, Abigail utiliza el siguiente ejemplo:

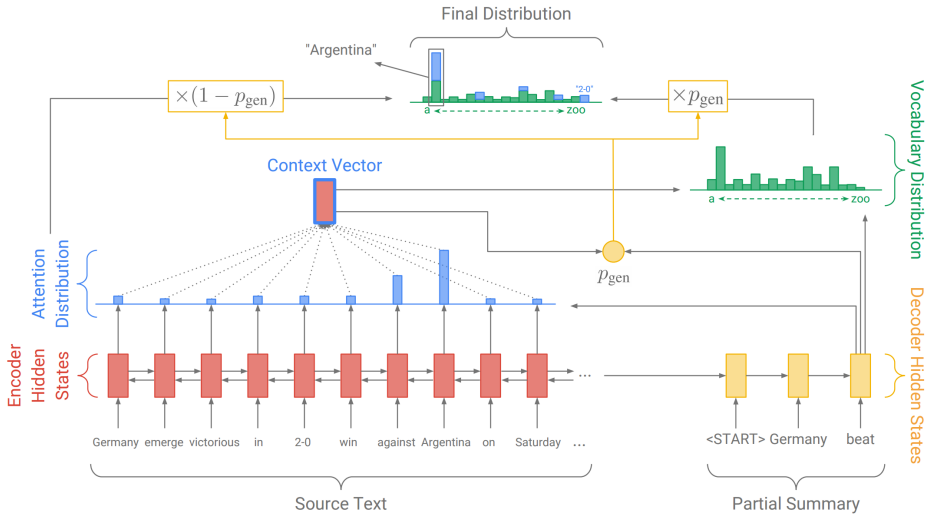

Figura 3.3: Ejemplo gráfico del funcionamiento de la red *pointer-generator* [14].

Lo primero que vemos en la base de la figura es la composición de la red *encoder* (celdas de color rojo) con cada una de las palabras del texto ejemplo, y de lo generado hasta el momento por la red *decoder* (celdas de color amarillo). Tanto el vector generado por el mecanismo de atención (el vector de distribución de atención), como la distribución del vocabulario que forma el contexto en el que nos encontramos, están siendo calculadas. Además de

[3]Repositorio de Github con el proyecto de Abigail See: `https://github.com/abisee/pointer-generator`

estos dos, se está calculando la probabilidad de que una palabra sea generada frente a que sea copiada desde el texto origen. Ésta, la llamada *probabilidad de generación* o p_{gen}, se usa para definir el peso de las distribuciones de atención y de vocabulario, y combinarlas en una única. Así, la distribución final nos dará la probabilidad con la que una palabra será generada, teniendo en cuenta su importancia dentro del contexto más la probabilidad de apuntar cualquiera de sus apariciones dentro del escrito origen.

Este modelo ofrece ventajas con respecto a las redes *sequence-to-sequence* tradicionales. El mecanismo de atención permite apuntar a una palabra en concreto si se necesita. Podremos entonces copiar del original al producido como si de un resumen *extractivo* se tratase. El hecho de poder apuntar o dirigir la atención a ciertas palabras, puede servir para mantener palabras de interés que no pertenezcan al vocabulario presente. Otra ventaja presente en el modelo *pointer-generator* es la rapidez que presenta aprendiendo.

El segundo problema a resolver era las numerosas repeticiones. Evitar aquello que ya está dentro del texto resultado o las repeticiones de conceptos se puede conseguir con la técnica llamada *coverage*. La cobertura se servirá de la distribución de atención de las distintas palabras del original para realizar un seguimiento de aquellos elementos que ya han sido cubiertos hasta el momento, y dejar de fijarse en otros ya repetidos decrementando su peso.

3.3. Generador de resúmenes: *Reinforcement Learning seq2seq*

Los modelos Cornerstone (3.1) y Pointer Generator con cobertura (3.2) tienen una cosa en común: sus redes presentan supervisión durante la etapa de entrenamiento, pero no durante la generación de los resultados. Esta supervisión dota a la red que forma el *encoder*, la capacidad de conocer aquello que tiene a su alrededor en la secuencia original; pero en cambio el *decoder* puede acumular errores conforme predice la siguiente palabra del resumen.

Romain Paulus [12] aborda en su artículo la implementación del aprendizaje por esfuerzo, y como con ella se pueden llegar a conseguir mejores resultados, no solo en métricas, sino también respecto al criterio humano. En su trabajo, Romain ofrece una alternativa a los modelos que tratan de redactar resúmenes calculando las palabras que minimizan la probabilidad de error a cada paso. Esta alternativa, *self-critic policy gradient training*, tiene como objetivo principal tomar decisiones que maximicen las futuras recompensas, de manera que las posibles métricas evaluadas sean las más óptimas posibles.

3.3.1. *Reinforcement Learning*

Esta idea de entrenamiento reforzado es algo muy importante en el campo del DL. El entrenamiento por refuerzo es una rama que pretende enseñar a una máquina a completar una tarea compleja mientras interactúa con el entorno en el que se encuentra. Las acciones que la máquina realiza dentro de este entorno para aprender son algo así como el aprendizaje de los niños. Cada acción tiene una recompensa o un castigo de manera que recuerde los pasos correctos hasta cumplir el objetivo.

Un ejemplo de la aplicabilidad de este método pueden ser las Inteligencias Artificiales (IA) que aprenden a jugar al ajedrez.

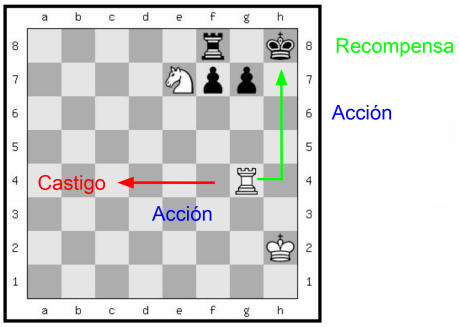

Figura 3.4: Aprendizaje por refuerzo de una IA en una partida de ajedrez.

Suponiendo que la IA de la [Figura 3.4] es el jugador blanco y que es su turno, la partida está casi decidida. Si la IA mueve la torre una casilla a la derecha, ganaría la partida por jaque mate. Si en cambio mueve en otra dirección o sentido perdería la oportunidad de ganar. Dependiendo de cuál sea su movimiento, recibirá una recompensa (la victoria) o un castigo (turno desaprovechado o pérdida de una figura propia).

3.3.2. Modelo *Reinforcement Learning* con *Policy Gradient*

Siguiendo lo propuesto por Romain, más tarde Yaser Keneshloo [7] analizó entorno al modelo *Pointer Generator*, diferentes implementaciones adicionales que ofrecen una mejora gracias al aprendizaje por esfuerzo. Entre estas implementaciones, se encuentra el modelo de *Policy Gradient*. Gracias a Yaser tenemos entre otras variantes el código[4] de este modelo para poder probarlo.

Lo que Romain y Yaser explican en sus artículos es la naturaleza de este

[4]Repositorio de GitHub con los distintos modelos ofrecidos por Yaser: https://github.com/yaserkl/RLSeq2Seq

aprendizaje y como afecta a los resultados. Mientras que en entornos fijos donde la red produce un texto de salida según una función que selecciona los elementos del resultado más convenientes a priori, en el entrenamiento con aprendizaje las decisiones se toman conforme a la política de selección actual. Esta política, o *policy*, es la función que determina el comportamiento del modelo, es decir, que acciones puede tomar. La recompensa se basará entonces en la diferencia que exista entre la frase recién generada y la secuencia base.

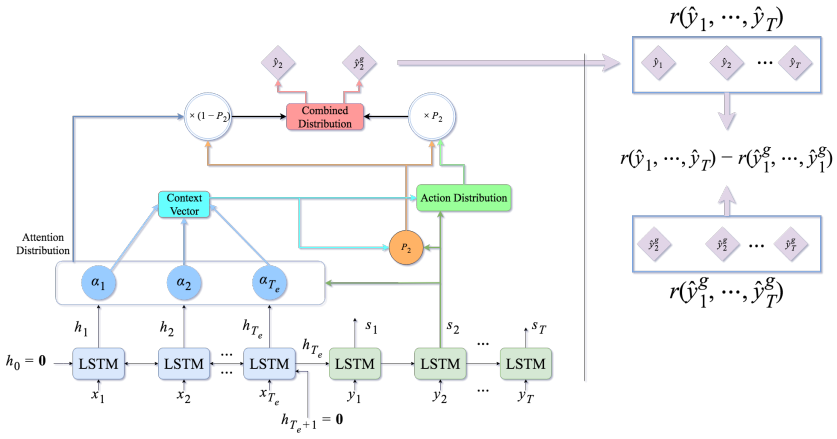

Figura 3.5: Ejemplo gráfico del funcionamiento de un modelo *Self Critic* basado en el *pointer generator seq2seq* con atención [7].

En la [Figura 3.5] se muestra como a cada paso del *decoder*, el vector del contexto es calculado a partir de su propio valor y la salida de esa capa combinados. Además gracias al *pointer generator*, esta distribución de acción (aprendida) se mezcla con la distribución de atención del *encoder* para obtener aún mayor seguridad en las predicciones. Tras esto se realiza el proceso anteriormente comentado, en el que se extrae del resultado generado, aquella salida base que se calculaba sin esta política de aprendizaje.

Por supuesto existen algunos problemas con este método. El hecho de ver la recompensa al finalizar la secuencia puede ser poco productivo. Si durante el proceso observamos que las recompensas no son las mejores, quizá sería conveniente cambiar de acción, cambiar de procedimiento para aprender mejor. Esto suele ocurrir en los inicios de la fase de entrenamiento, cuando las decisiones y predicciones que se toman son aún ineficientes y arbitrarias. Pero existen dos maneras de evitar en gran medida esto. La primera es preentrenar el modelo para más tarde hacer el cambio y pasar a un aprendizaje por refuerzo. Y la segunda opción sería prestar más importancia a las palabras recién generadas para la producción de nuevas en el resumen, en lugar de depender de las *tokens* del artículo original [2].

CAPÍTULO 4

Resultados

4.1. Colecciones de referencia utilizadas

La colección de artículos usada para entrenar y producir titulares con la implementación del modelo *Cornerstone seq2seq* es el corpus de Gigaword. Esta colección reunida por el *Linguistic Data Consortium* (LDC) [3], contiene alrededor de 4 millones de artículos en inglés, recogidos de distintos periódicos de medios de información norteamericanos.

Las colecciones usadas para las dos implementaciones que generan resúmenes, el *Pointer Generator* y *Reinforcement Learning seq2seq*, son artículos de las cadenas de noticias CNN y Daily Mail. Estos artículos fueron recogidos por Karl M. Hermann [5], para enseñar a las máquinas a leer y comprender documentos del lenguaje natural. Las colecciones contienen un total de 90 mil y 180 mil documentos de cada una.

4.2. Resultados *Cornerstone seq2seq*

Después de haber analizado la estructura del modelo para la generación de titulares, el siguiente paso es probar su capacidad. Tras preparar el modelo para su uso, hay que entrenarlo. Partiendo de los parámetros base, una de las mejores opciones para evaluar su potencia y precisión es cambiar los parámetros y realizar distintas pruebas. En cada ocasión se entrenará un número de pasos determinados y variando a su vez también características como la profundidad de la red y el número de capas entre otras.

Los resultados obtenidos al evaluar los distintos modelos en base a las métricas ROUGE, concretamente con la versión de *python* pyROUGE, se muestran en la [Tabla 4.1].

Prueba	Métrica evaluada	Recall	Precision	medida-F
Modelo nº1	ROUGE-1	0.37203	0.46633	0.40260
	ROUGE-2	0.16547	0.21068	0.17946
	ROUGE-L	0.34892	0.43661	0.37720
Modelo nº2	ROUGE-1	0.32645	0.43166	0.35995
	ROUGE-2	0.14121	0.19091	0.15631
	ROUGE-L	0.30720	0.40636	0.33863
Modelo nº3	ROUGE-1	0.25083	0.34899	0.28236
	ROUGE-2	0.08726	0.12572	0.09874
	ROUGE-L	0.23428	0.32625	0.26364

Tabla 4.1: Resultados ROUGE obtenidos sobre los resúmenes producidos por el generador de artículos.

Como ya se ha comentado, lo ideal para evaluar como trabaja el generador de artículos es probar diferentes variaciones del modelo [Tabla 4.1]. Con el primer modelo con el que se trabajó, se optó por utilizar a modo de prueba los *word embedding* y el modelo pre-entrenado que nos ofrece la herramienta de Glove. Para el segundo y tercer modelo se entrenaron de forma manual. El segundo modelo fue entrenado con una red de tamaño 200 celdas, y que ha realizado 16000 pasos, en los que la red ha ido actualizando sus parámetros internos. Para el modelo número 3 se optó por ampliar la extensión de la red. Con un tamaño de 300 nodos y 4 capas, se entrenó otros 16000 pasos.

Algunos de los ejemplos obtenidos de forma satisfactoria son los que se observan en tablas [Tabla 4.2] y [Tabla 4.3]. Podemos comprobar que los titulares obtenidos con los modelos coinciden en gran medida, no solo en las palabras que contienen, sino en el sentido de lo que expresan. No habiendo lugar a duda que leyendo cualquiera de los titulares se comprende fácilmente los que los textos originales cuentan. Queda claro entonces que los resultados generados son totalmente aceptables, a pesar de que los modelos presentan diferentes configuraciones en sus estructuras, y del corto tiempo de entrenamiento que han realizado.

En particular, en el segundo ejemplo ([Tabla 4.3]), que habla sobre los casos de gripe aviar en Turquía, se han obtenido titulares que transmiten la misma idea que el de referencia. El titular producido por el modelo nº1 ofrece la información incluso más precisa que el titular usado de referencia, estableciendo de manera más concreta la región en la que se produjo el hecho.

A continuación, el siguiente ejemplo ([Tabla 4.4]) muestra un grado mayor

Titular de referencia:
arsenal boss fears losing henry
Titular modelo nº1:
arsenal fears henry to leave
Titular modelo nº2:
gunners worry henry to leave arsenal
Titular modelo nº3:
henry to quit at arsenal

Tabla 4.2: Ejemplo nº1 comparando los resúmenes generados con respecto al titular de referencia.

Titular de referencia:
two test positive for bird flu virus in turkey
Titular modelo nº1:
two people tested positive for bird flu in eastern turkey
Titular modelo nº2:
at least two tests positive for bird flu
Titular modelo nº3:
two tests positive for bird flu

Tabla 4.3: Ejemplo nº2 comparando los resúmenes generados con respecto al titular de referencia.

de imprecisión a la hora de resumir el artículo.

Mientras que el modelo número 1 refleja en buena medida lo que dice el texto original, coincidiendo con el titular de referencia, en el caso del segundo modelo se aprecia un cambio total en el sentido del artículo. Este titular llega a indicar que las víctimas del tráfico ilegal de personas serán deportadas, en lugar de tener la oportunidad de permanecer de forma permanente en el país.

A pesar de todo esto, se han encontrado casos en los que los modelos han sido incapaces de reproducir el mismo éxito al resumir. Algunas de las causas por las que esto puede haber ocurrido van desde el poco entrenamiento de las redes, hasta no contar con las palabras precisas o suficientes en el vocabulario. En cualquiera de estos casos, el *decoder* coloca una *token* <UNK> para reflejar que no la contempla en su contexto o que no es capaz de encontrar un palabra cuya probabilidad de similitud asegure que sea la correcta para formar parte del resumen. Un ejemplo de esto queda representado en la [Tabla 4.5].

Se puede ver como los modelos, por aparente falta de vocabulario, no han logrado sintetizar del todo el artículo sobre una nueva especie de planta. Aunque en este ejemplo un gran porcentaje de la frase si ha sido simplificada,

Titular de referencia:
human trafficking victims could get right to remain in britain
Titular modelo nº1:
britain bans deportation of human trafficking
Titular modelo nº2:
human trafficking may be deported from britain
Titular modelo nº3:
human rights group says it may be married

Tabla 4.4: Ejemplo nº3 comparando los resúmenes generados con respecto al titular de referencia.

Titular de referencia:
malaysia probes possible new species of world 's largest flower
Titular modelo nº1:
new species of <unk > found in malaysia
Titular modelo nº2:
experts find new species of
Titular modelo nº3:
experts find <unk >

Tabla 4.5: Ejemplo nº4 comparando los resúmenes generados con respecto al titular de referencia.

en otros titulares producidos la relación es peor. Desde titulares vacíos, con una sola *token* <UNK> e incluso incompletos.

4.3. Resultados *Pointer Generator*

Habiendo visto el enfoque con el se intenta resolver la tarea, es tiempo de probar el generador de resúmenes y ver los resultados que se pueden llegar a obtener si basamos el procedimiento en extraer la información e ideas más relevantes y manteniéndolas en el foco.

Antes de entrar en detalle y analizar los resultados, es necesario comentar algunas características con respecto a la implementación. Puesto que el proyecto original está construido sobre una versión más antigua de *Python* y *Tensorflow*, lo primero es crear un entorno capaz de albergar estas particularidades. El usuario Becxer ofrecía una versión algo más actualizada de *Python* en su repositorio de GitHub [1], así que, se decidió partir de está. Además, y debido a las complicaciones encontradas para hacer funcionar el

[1]Bifurcación de Becxer sobre el código original del Pointer Generator: https://github.com/becxer/pointer-generator/

código, se optó por utilizar *Jupyter notebooks*. Concretamente, y de manera simple, se recurrió a los cuadernos que se pueden desarrollar en Google Colaboratory, tal como Amr Zaki [17] realizó probando modelos para la generación de resúmenes. Esto supone entonces una forma cómoda de observar la implementación del modelo, y disponer de un entorno rápido y potente con el que reproducir el experimento a través de Internet.

Tras haber puesto en situación el contexto sobre el que se aloja el modelo, hay que hablar sobre su estructura. Algunos de los parámetros base que dan forma a las redes neuronales de este modelo son las siguientes:

- Una red neuronal recurrente de dimensión 256 celdas.
- El número de pasos para la fase de *encode* es de 400. Esto significa que los textos de entrada tendrán un máximo de 400 *tokens* o palabras.
- El número de pasos con los que se producirán los elementos del resumen en el *decoder* son 100. Con esto se consigue que los resúmenes generados tengan un máximo de 100 palabras.
- La longitud mínima de los resúmenes generados, para el *beam search decoding mode*, es de 35.

Además, contamos con dos parámetros o *flags* adicionales que permiten al usuario decidir si usar el modelo *pointer-generator* o no, y si aplicar o no el método de cobertura como mecanismo de atención. Los resultados ROUGE que han sido obtenidos en la evaluación son los que se pueden observar en la [Tabla 4.6], y coincidiendo con el estudio original de Abigail, las métricas obtenidas con el modelo que cuenta con cobertura son muchos mejores.

	ROUGE-1	ROUGE-2	ROUGE-L
Pointer Generator	0.25402	0.10008	0.18569
Pointer Generator + Coverage	0.39150	0.17210	0.28764

Tabla 4.6: Resultados ROUGE obtenidos sobre los resúmenes producidos por el generador de resúmenes: Pointer Generator.

La primera prueba fue realizada sólo con *pointer-generator*. No se contó con ningún mecanismo de atención para ayudar al modelo. En la segunda prueba del experimento si se llevó a cabo con el mecanismo de cobertura para observar su capacidad y las diferencias que pudiesen surgir. Estos son dos ejemplos de los artículos originales junto a los resúmenes de referencia y generados por el modelo:

Artículo original (truncado):

england 's campaign at the european under 21 championship looks set to be dominated over whether harry kane is included - but gareth southgate may not need him . in front of over 30,000 at the riverside stadium , england came from behind to defeat germany 3-2 and record a victory against not just old rivals but the pre-tournament favourites in the czech republic this summer . southgate is still expected to do everything he can to include the tottenham striker , but the win over germany shows that the young lions can go toe-to-toe with the cream of europe . here sportsmail looks at england 's three stand-out stars from the win in middlesbrough . james ward-prowse . the southampton star has already become an established member of the first team at st mary 's and is set to be a key member of southgate 's side in the czech republic . the 20-year-old scored the winner with a crisp low effort after _ _ghosting_ _ into the penalty box to find space . but his attacking play extends to set-pieces . germany 's julian _ _korb_ _ reaches out in vain as james ward-prowse fires home england 's winning goal . the under 21 captain celebrates his late strike following a fine midfield performance . in ward-prowse , england have potentially their best _ _dead-ball_ _ specialist since david beckham with his passing range and accurate shooting already a highlight of his game . ward-prowse was captain for the germany victory and his rapid progress on the south coast in the last two years suggests he is worth keeping an eye on . carl jenkinson . named the official man of the match , jenkinson put in a close to perfect performance - albeit he was slightly at fault for germany 's opening goal after allowing the cross for philipp _ _hofmann_ _ to fire home . but after a slow start to each half , the west ham full-back , on loan from arsenal , imposed himself down the right-hand side . carl jenkinson runs the ball out of defence for england as germany substitute nico schulz looks on . jenkinson put in a man of the match display having assisted two of england 's three goals . it was the 23-year-old 's cross for jesse lingard which helped draw the three lions level in the first half , while he also provided the pass for ward-prowse to tuck home the winning goal . with england 's senior team lacking depth in the right-back area , roy hodgson could do worse than pick jenkinson . because if he does n't , then finland will as he can still play for the scandinavian team . nathan redmond . relished the battle with germany full-back christian gunter , often managing to pull away from his marker to lead an attack down the right wing . redmond is not all about pace though as he often tried to cut inside to mix up his approach , with his passing in the final third managing to catch the german back four flat on a number of occasions . nathan redmond rifles home england 's second equaliser across goal into the bottom corner . redmond celebrates after drawing england level late in the game following an impressive display on the right . it was a much improved version of the winger who struggled in a poor norwich side in the premier league last season , but having struck the equaliser against a talented germany team he looks to be improving ...

Resumen de referencia:
england recorded impressive 3-2 win over germany in under 21 friendly . young lions twice came from behind to seal late win at riverside stadium . james ward-prowse , carl jenkinson and nathan redmond among stars . gareth southgate 's side look in strong shape heading to the european under 21 championship in czech republic this summer .
Resumen modelo Pointer Generator:
european under 21 championship looks set to be dominated over whether harry kane . the 20-year-old scored the winner with a crisp low effort after ghosting into the penalty box to find space . the 20-year-old scored the winner with a crisp low effort after ghosting into the penalty box to find space .
Resumen Pointer Generator + mecanismo de atención de Cobertura:
in front of over 30,000 at the riverside stadium , england came from behind to germany 3-2 and record a victory against not just old rivals but the pre-tournament favourites in the czech republic this summer . southgate is still expected to do everything he can to include the tottenham striker .

Tabla 4.7: Ejemplo nº1 comparando los resúmenes generados por PG con respecto al artículo original y el resumen de referencia.

Una de las principales características de las que se habló cuando se describió el modelo fue la capacidad de poder copiar y recuperar palabras clave que quizá no entrasen en el vocabulario debido a su rareza y escasez. Se puede observar aquí como palabras, comúnmente sustantivos propios, tales como "Kane"(apellido del joven futbolista) o "Southgate"(apellido del entrenador de la selección) aparecen en los resúmenes generados gracias a esa capacidad de .ªpuntar" a palabras con importancia dentro del contexto.

Otro de los principales problemas que se pretendía resolver añadiendo el mecanismo de cobertura al modelo, era la repetición de frases en el resumen producido. Y es en el primer modelo donde se ve de manera clara (frase subrayada), como el *decoder* volvió a centrar su atención en la importancia del gol del jugador veinteañero. Esto no debería suceder, como en el resumen producido por el *pointer-generator* con cobertura, donde no se encuentra repetición alguna.

El segundo ejemplo a analizar será el siguiente:

Artículo original (truncado):

rangers boss stuart mccall says he feels sorry for newcastle loanee gael __bigirimana__ , but admits he should never have been signed by the ibrox club . __burundi-born__ england youth cap __bigirimana__ has yet to feature for the light blues after being diagnosed with a mystery illness . he was one of five magpies youngsters signed by former light blues chief executive derek llambias without being put through a medical - even though __bigirimana__ knew he was sick before signing for the glasgow giants . gael __bigirimana__ has not played for rangers since his loan move from newcastle . the 21-year-old has denied rumours claiming he is suffering from __hepatitis-c__ , but __gers__ boss mccall confirmed his condition means he will not be able to feature for his adopted club . the former motherwell manager would love to have the use of the former coventry midfielder 's services , but believes he should never have been allowed to make the move north in the first place . ‘ __bigi__ has got a medical condition which is a personal matter , 'said mccall . ‘ the doctor and consultant are all involved in that . í knew him at coventry . i remember watching him and thought when i came into the job he will be a good one to bring in and give us a bit of energy . unfortunately because of his medical condition he wo n't be able to play for us . __bigirimana__ signed on loan for rangers alongside haris __vuckic__ -lrb- middle -rrb- and __remie__ __streete__ . ‘ but that is as big a blow to the kid as it is for us . it 's not his fault , this -lsb- row -rsb- has been nothing to do with him . he comes in with a smile on his face each morning , he trains with the fitness coach as hard as he possibly can . ‘ so as disappointed as we are that we ca n't use him , you have to think of the boy in this matter because he ca n't go back , he ca n't play for anyone else . ‘ it 's an unfortunate circumstance that really should never have happened . 'llambias - a close associate of magpies owner mike ashley - signed off on the five newcastle loan deals just hours before the january transfer window shut , but did not ask club doctors to check the players over . so far only slovenian playmaker haris __vuckic__ has played regularly . __bigirimana__ says he knew he was ill before he moved to rangers . defender __remie__ __streete__ limped off just half an hour into his debut and has not been seen since , while northern ireland winger shane ferguson has yet to even step foot inside murray park after being ruled out for the rest of the season with a serious knee injury . swiss defender kevin __mbabu__ has featured for rangers 'youth side but is ‘ nowhere ñear being fit enough for first-team duty , according to mccall . the __gers__ boss cans sympathise with the players but hit out at the previous regime 's decision to sanction some of the signings . rangers boss stuart mccall says he feels sorry for newcastle loanee __bigirimana__ . mccall said : ‘ should these guys ever have been sent here ? well , with __bigi__ being unable to play that 's a no-brainer . with shane , he was on his way back but had a little setback ...

Resumen de referencia:
stuart mccall says gael __bigirimana__ should n't have been signed by rangers . __bigirimana__ has yet to feature since signing on loan from newcastle . the 21-year-old midfielder has been diagnosed with a mystery illness .
Resumen modelo Pointer Generator:
gael bigirimana has not played for rangers since his loan move from newcastle . he was one of five magpies youngsters signed by former light blues chief executive derek llambias . he was one of five magpies youngsters signed by former light blues chief executive derek llambias without being put through a medical - even though bigirimana knew he was sick before signing for the glasgow giants .
Resumen Pointer Generator + mecanismo de atención de Cobertura:
newcastle loanee gael bigirimana has not played for rangers since his loan . gael bigirimana admits he should never have been signed by the ibrox club . the 21-year-old has denied rumours claiming he is suffering from hepatitis-c .

Tabla 4.8: Ejemplo nº2 comparando los resúmenes generados por PG con respecto al artículo original y el resumen de referencia.

En esta segunda prueba se encuentran de nuevo evidencias de como el modelo es capaz de copiar y extraer ciertas palabras que claramente contienen importancia dentro de las noticias y puede que no estén dentro del vocabulario. Igualmente, se observa otra vez como en el texto generado por el modelo que carece de mecanismo de atención, aparece la misma frase dos veces. En este caso, al contrario que en el anterior ejemplo, el *decoder* repite la misma oración pero amplia su significado completando con más elementos.

En definitiva, aunque es obvio que sólo mostrando dos ejemplos que han funcionado no se demuestra la eficacia del generador de resúmenes, se infiere que las mejoras que ofrece este modelo son de gran ayuda. Poder evitar la repetición y el olvido de palabras relevantes pero no comunes es un gran logro con el que seguir adelante hasta conseguir resúmenes "humanizados". Aún hay matices de los cuales Abigail habla en su *blog*, que se pueden ver a simple vista en los resúmenes producidos. Muchas de las frases reproducen prácticamente palabra por palabra las frases sobre las que se fija en el escrito original. Esto se acerca más a esa forma *extractiva* de la que se habló anteriormente. Otro de los objetivos que podrían ayudar a conseguir aún mejores resultados, es el conseguir mantener una idea más general en aquellos resúmenes con más de una frase, pues es cierto que se han dado casos en los que las distintas partes del resultado generado no guardan aparente relación entre sí aunque se hayan extraído del mismo artículo.

4.4. Resultados *Reinforcement seq2seq*

El modelo de *Reinforcement Learning* con *Policy Gradient* es una extensión del modelo de Abigail See que tan buenos resultados ha dado. Contando sus redes con las mismas estructuras que las del *Pointer Generator*, sus parámetros y características son muy similares.

Al igual que en el modelo anterior, se decidió optar por establecer la implementación de este código en un entorno potente como el de Google Colaboratory. Gracias de nuevo al trabajo de Amr Zaki [17], la adaptación fue rápida. Sólo quedaría actualizar el código a *Python 3* y agrupar e instalar todo lo necesario. Una vez todo se haya establecido, lo ideal es realizar dos fases dentro del entrenamiento. Una primera fase servirá de pre-entrenamiento para los parámetros de la red. Después, en la segunda fase, se pasará a entrenar el modelo para que aprenda de acuerdo al *Policy Gradient*. En total se realizó un entrenamiento de más de 15k pasos. Las métricas obtenidas en la evaluación con los dos modelos se muestran en la [Tabla 4.9]. En ella se ve reflejado como los resultados de las mediciones son mejores que en el modelo básico *pointer-generator*. El objetivo principal del aprendizaje por refuerzo, producir resultados con métricas elevadas, parece haberse cumplido.

	ROUGE-1	ROUGE-2	ROUGE-L
Pointer Generator	0.22153	0.08000	0.19327
PG con RL+Policy Gradient	0.24558	0.10261	0.21464

Tabla 4.9: Resultados ROUGE obtenidos sobre los resúmenes producidos por el generador de resúmenes: Reinforcement Learning seq2seq.

Una de las primeras cosas que se pueden observar entre los resultados, es la aparición de repeticiones y fallos identificando el sujeto de la oración. Esto se debe a que ambos modelos carecen del mecanismo de atención de cobertura que presentaba el modelo de Abigail y que evitaba esta clase de problemas. A pesar de esto, y como se ve reflejado en los artículos originales, la legibilidad y coherencia de los textos producidos es notable.

El ejemplo de la [Tabla 4.10] ofrece una idea de esto que se acaba de comentar. Se dan repeticiones, pero en su gran mayoría los resúmenes son coherentes y siguen hilos bastante naturales. No se producen frases independientes con ideas que apenas se conectan entre ellas. Así se consigue una lectura completa del artículo original.

Artículo original (truncado):
marseille , france -lrb- cnn -rrb- the french prosecutor leading an investigation into the crash of __germanwings__ flight __9525__ insisted wednesday that he was not aware of any video footage from on board the plane . marseille prosecutor brice robin told cnn that " so far no videos were used in the crash investigation . " he added , " a person who has such a video needs to immediately give it to the investigators . " robin 's comments follow claims by two magazines , german daily bild and french paris match , of a cell phone video showing the harrowing final seconds from on board __germanwings__ flight __9525__ as it crashed into the french alps . all 150 on board were killed . paris match and bild reported that the video was recovered from a phone at the wreckage site . the two publications described the supposed video , but did not post it on their websites . the publications said that they watched the video , which was found by a source close to the investigation . " one can hear cries of ' my god ín several languages , " paris match reported . " metallic banging can also be heard more than three times , perhaps of the pilot trying to open the cockpit door with a heavy object . towards the end , after a heavy shake , stronger than the others , the screaming intensifies . then nothing . " " it is a very disturbing scene , " said julian __reichelt__ , editor-in-chief of bild online . an official with france 's accident investigation agency , the bea , said the agency is not aware of any such video . lt. col. jean-marc __menichini__ , a french __gendarmerie__ spokesman in charge of communications on rescue efforts around the __germanwings__ crash site , told cnn that the reports were " completely wrong " and " unwarranted . " cell phones have been collected at the site , he said , but that they " had n't been exploited yet . " __menichini__ said he believed the cell phones would need to be sent to the criminal research institute in __rosny__ __sous-bois__ , near paris , in order to be analyzed by specialized technicians working hand-in-hand with investigators . but none of the cell phones found so far have been sent to the institute , __menichini__ said . asked whether staff involved in the search could have leaked a memory card to the media , __menichini__ answered with a __categorical__ " no . " __reichelt__ told " erin burnett : outfront " that he had watched the video and stood by the report , saying bild and paris match are " very confident " that the clip is real . he noted that investigators only revealed they 'd recovered cell phones from the crash site after bild and paris match published their reports . " that is something we did not know before . overall we can say many things of the investigation were n't revealed by the investigation at the beginning , " he said . what was mental state of __germanwings__ co-pilot ? german airline lufthansa confirmed tuesday that co-pilot andreas __lubitz__ had battled depression years before he took the controls of __germanwings__ flight __9525__ , which he 's accused of deliberately crashing last week in the french alps . __lubitz__ told his lufthansa flight training school in 2009 that he had a " previous episode of severe depression , " the airline said tuesday ...

Resumen de referencia: marseille prosecutor says “ so far no videos were used in the crash investigation ” despite media reports . journalists at bild and paris match are “ very confident ” the video clip is real , an editor says . andreas !!__lubitz__!! had informed his lufthansa training school of an episode of severe depression , airline says .
Resumen modelo Pointer Generator: marseille prosecutor brice robin told cnn that “ so far no videos were used in the crash investigation . the two publications described the video was recovered from a phone at the wreckage site .
Resumen modelo Reinforcement Learning + Policy Gradient: marseille prosecutor brice robin prosecutor leading an investigation into the crash of germanwings flight 9525 . he says he was not aware of any video footage . the cell phones have been sent to the investigation .

Tabla 4.10: Ejemplo comparando los resúmenes generados por RL seq2seq con respecto al artículo original y el resumen de referencia.

Si nos fijamos más detalladamente el los resúmenes generados en ambos experimentos podemos observar una característica principal del *pointer-generator*. Mientras que el modelo entrenado con RL también cuenta con la estructura que le da la opción de extraer pequeños trozos del artículo original, reorganiza la oración sin copiar palabra por palabra del escrito original. Presenta un carácter más *abstractivo* que los resultados reproducidos por el modelo que no usa aprendizaje por refuerzo.

También, destacar otro de los fallos que más ha ocurrido con RL y *Policy Gradient*: la secuencia subrayada en amarillo no solo es símbolo de las repeticiones ya comentadas anteriormente, sino que, tanto en éste como en gran cantidad de los resúmenes producidos se han encontrado frases cortas que carecían de sentido o daban la imagen de no haber sido completadas. Esto coincide con los resultados obtenidos en el proyecto original de Romain, donde también se afirma que, a pesar de esto, los índices de legibilidad son los más altos en comparación a otros experimentos.

CAPÍTULO 5

Conclusiones y trabajo futuro

5.1. Conclusiones

El trabajo comenzó con el estudio de las bases del *Deep Learning*. Estas bases comprendían técnicas y métodos para la solución de tareas de manera automática. Entre estos métodos se encontraban aquellos más adecuados para la realización de mi objetivo: la generación de resúmenes de forma automatizada. Modelos de redes neuronales, tipos de celdas, representación de las palabras, estructuras *encoder-decoder* y los mecanismos de atención fueron las características principales estudiadas y son los puntos más importantes dentro del procesamiento del lenguaje natural.

Después de este estudio previo, se buscaron proyectos punteros en el campo para poder reproducir sus experimentos y analizar los resultados. Se recurrió a tres modelos que ofrecían distintos enfoques y contaban con diferentes estructuras en sus implementaciones. El modelo más fundamental, el generador de titulares *Cornerstone seq2seq*, fue configurado y ejecutado de manera simple en un ordenador personal. Por otro lado, los modelos *Pointer Generator* y *Reinforcement Learning seq2seq* fueron adaptados a cuadernos Jupyter ejecutados en la herramienta *Google Colaboratiry*. Esta herramienta proporcionó una alternativa rápida, eficiente y cómoda de utilizar debido a los problemas de incompatibilidad y durante el funcionamiento en el contenedor disponible en la universidad.

Según los resultados obtenidos lo primero que se debe aclarar es el objetivo de los distintos tipos de implementaciones. El alcance de cada una requiere mayor profundidad y complejidad en su estructura para poder lo-

grar resultados significativos. El generador de titulares es un claro ejemplo de una estructura básica, busca la simplificación de artículos para poder redactar en una sola frase, un titular que capte la idea del texto original. Si se comparasen las frases, de manera individual, entre las generadas por titulares y las que comprenden los resúmenes más extensos de los otros dos modelos, se observa que todas son coherentes. Los titulares al tener menor longitud en sus secuencias, contienen frases que incluyen menos detalles. Se puede afirmar que el modelo *Cornerstone* es capaz de simplificar los escritos originales imitando los titulares de una noticia escrita por periodistas.

Las comparaciones que se pueden establecer entre los generadores de resúmenes son mayores. Sus bases son las misma por lo que tienen más cosas en común. Pueden llegar a reproducir varias frases transmitiendo así una idea más clara de aquello que se pretende resumir. Ambos modelos adoptan el concepto de *Pointer Generator* y parten en diferentes direcciones según los problemas que tratan de resolver. El modelo *Pointer Generator* se sirve además del mecanismo de atención llamado *coverage* o cobertura para tratar de evitar las repeticiones; mientras que el modelo que recurre al aprendizaje por refuerzo, *Reinforcement Learning*, utiliza el *Policy Gradient* para maximizar las recompensas durante su entrenamiento.

Mientras que el modelo PG con cobertura logra evitar las repeticiones, pues marca las zonas donde ya ha extraído palabras y conceptos, los resultados obtenidos por el modelo de RL si presentan secuencias repetidas. Al igual que pasaba en la generación de resúmenes sin mecanismo de atención, la relevancia de ciertas palabras en el artículo concentra demasiada atención, haciendo que el *decoder* las tome en cuentas varias veces. Por el contrario, algo que el modelo RL consigue es evitar ese carácter *extractivo* que caracteriza al PG. Mientras que este aprende a elegir las palabras que garanticen una mejor secuencia resultado, el PG se enfoca algo más en localizar puntos de gran importancia y parafrasearlos. Si bien es cierto que en ambos casos, los modelos cumplen su función y crean sentencias totalmente legibles y de calidad, dan impresiones diferentes al lector.

Por último, puntualizar que a pesar de haber medido los valores ROUGE de las tres implementaciones, los resultados de las métricas quedan por debajo de los conseguidos en los artículos originales de sus autores. Esto puede deberse a la cantidad de entrenamiento que han recibido durante este proyecto. Debido a las dificultades que han surgido durante los procesos de configuración y de ejecución, los tiempos de entrenamiento han sido mucho menores que los completados durante su desarrollo original. Algo que ha quedado claro es que a mayor tiempo de entrenamiento, mejores resultados se obtienen.

5.2. Trabajo futuro

Cada pocos años surgen nuevos enfoques que intentan solucionar la difícil tarea que es conseguir resúmenes abstractivos. Una posibilidad para ampliar este trabajo en el futuro es la búsqueda de nuevos modelos que puedan llegar a obtener mejores resultados agregando nuevas técnicas *Deep Learning*.

Uno de los modelos a examinar fue desarrollado por Yaser Keneshloo [7] basándose en el modelo propuesto por Samy Bengio [2]. En este se pretende evitar la tendencia de los modelos a depender de las *tokens* previas del texto original y en cambio focalizar la influencia de las palabras recientemente generadas en los pasos anteriores.

Otro proyecto desarrollado recientemente en 2020 utiliza *transformadores*. Estos modelos con estructura *encoder-decoder* han ganado mucha fama, pues son más efectivos a la hora de modelar las dependencias presentes en largas secuencias de palabras. *PEGASUS* [18], que son las siglas de este proyecto, diseñado por Jingqing Zhang, tiene como objetivo que el modelo aprenda por sí mismo, habiéndole ocultado previamente partes importantes del los textos de entrada para que sean generadas como resultado. Esto ha demostrado resultados sorprendentes y sería un excelente ejemplo a analizar y comparar.

Bibliografía

[1] Bahdanau, D., Cho, K. y Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *Conference paper at ICLR 2015*, 2016.

[2] Bengio, S., Vinyals, O., Jaitly, N. y Shazeer, N. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *arXiv e-prints*, página arXiv:1506.03099, 2015.

[3] Graff, D., Kong, J., Chen, K. y Maeda, K. English gigaword. *Linguistic Data Consortium, Philadelphia*, vol. 4(1), página 34, 2003.

[4] He, R., Lee, W. S., Ng, H. T. y Dahlmeier, D. An unsupervised neural attention model for aspect extraction. En *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, páginas 388–397. Association for Computational Linguistics, Vancouver, Canada, 2017.

[5] Hermann, K. M., Kociský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M. y Blunsom, P. Teaching machines to read and comprehend. *CoRR*, vol. abs/1506.03340, 2015.

[6] Hochreiter, S. y Schmidhuber, J. Long Short-Term Memory. *Neural Computation 9(8):1735-1780*, 1997.

[7] Keneshloo, Y., Shi, T., Ramakrishnan, N. y Reddy, C. K. Deep reinforcement learning for sequence to sequence models. *CoRR*, vol. abs/1805.09461, 2018.

[8] Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. En *Text Summarization Branches Out*, páginas 74–81. Association for Computational Linguistics, Barcelona, Spain, 2004.

[9] Loye, G. Attention mechanism. 2020.

[10] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. y Dean, J. Distributed representations of words and phrases and their compositionality. *CoRR*, vol. abs/1310.4546, 2013.

[11] Olah, C. Understanding LSTM Networks. 2015.

[12] Paulus, R., Xiong, C. y Socher, R. A deep reinforced model for abstractive summarization. *CoRR*, vol. abs/1705.04304, 2017.

[13] Pennington, J., Socher, R. y Manning, C. D. Glove: Global vectors for word representation. En *Empirical Methods in Natural Language Processing (EMNLP)*, páginas 1532–1543. 2014.

[14] See, A., Liu, P. J. y Manning, C. D. Get to the point: Summarization with pointer-generator networks. *CoRR*, vol. abs/1704.04368, 2017.

[15] Wang, Y., Huang, M., Zhu, X. y Zhao, L. Attention-based LSTM for aspect-level sentiment classification. En *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, páginas 606–615. Association for Computational Linguistics, Austin, Texas, 2016.

[16] Zaki, A. Beam search & attention for text summarization made easy (tutorial 5). 2019.

[17] Zaki, A. M., Khalil, M. I. y Abbas., H. M. Deep architectures for abstractive text summarization in multiple languages. En *14th IEEE International Conference on Computer Engineering and Systems (ICCES 2019)*. IEEE, Cairo, Egypt, 2019.

[18] Zhang, J., Zhao, Y., Saleh, M. y Liu, P. J. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *ArXiv*, vol. abs/1912.08777, 2019.